AUGUST '84

I-M 1 IN A MILLION CLUB

NATIONAL NEWSLETTER

Look! Up on the hill! It's a bug! It's a ... No! It's...
SPEED! CENTIPEDE! CENTIPEDE! CENTIPEDE! CENTIPEDE! CEN
The popular ATARI arcade game is here for your ATARI!
Now $9.95

LIGHT-TRACER

Maneuver the "light tracer" within the screen boundaries.
Now $4.95

THREE-FOR-ONE

1)SPACE WARD—You pilot an armored battle in outer space.
2)AIR FORTRESS—Defend the fortress from the kamakaze aliens.
3)BOMBER—Protect every block moving to destroy the base!
All three now $9.95

THE SEARCH

Explore the land in this text adventure game. Find the magic
sword and hunt for the villain.) The path is not easy but
fun and full of surprises. Once you beat Mother Nature and other
enemies, it's an adventure that uses two-word commands which
you must type in.
Only $9.95

****

SHOOT-OUT

Hunt down and destroy the enemy in the different sectors of
the galaxy. All commands are three letters. Special features
are used in this game. Each of your weapons are handled now.
You'll get a current view of the sector. If your lasers are dam-
aged you lose half of your firepower. If your enemies are de-
stroyed you are only rated on how? special position for
those with bonus discs.
Only $9.95

A Sequel from G.M.! Soon for CENTIPEDE! is?
"Contributions on Chris land Game"

Best of all you can learn to COMPUTER light. Have you got a
single program idea that could build skill? Send it to us!
You will be given credit for it and get paid for it for the
it. It will be advertised at distributed through us. Royalty
pay will be disclosed at negotiation time anytime else. All sub-
mitted ideas will be subject to some changes.

GENERAL NEWS

SPECIAL THANKS!

To BILL BOWMAN, LOUIS DOOLEY, and HAL BLOOM for supplying the club with programs that appear in this issue and in the CLUB PROGRAM LIBRARY.

Listed below is a brief explanation of each program submitted.

BILL BOWMAN—A program that will compare the cost of fuel--OIL vs GAS vs ELECTRIC. Club program designation 76FUEL20MT.

HAL BLOOM—A HARD COPY ONLY of a musical program that contains 14 different tunes that is set up to full sort tune from disk and RUN it. This program is a LIBRARY SELECTION but IS NOT on tape or disk. Eleven pages will be reproduced and sent to you, and counted as ONE selection. The minimum library order is 3, so your other two selections will be sent on cassette. Library designation HCDISKMUS

LOUIS DOOLEY-"APP IM-1 TYPEWRITER Jr."
This program allows the user to type data, messages, records, etc. to the screen and store many screens to tape. Reviewing each screen is done by simply tapping the space bar. The man explains the functions of the program, Bk. Designation 76TYJA.

BILL BOWMAN-A program that will indicate HOW LONG a fixed sum of money, earning a fixed interest rate, would last at a specified monthly withdrawal. Designation 76TIMAOL.

CORRECTION TIME!

An error (typo) that appeared in the JULY ARCADE is a good example of what happens when EXTREME CARE is not taken when LISTING or ENTERING ML DATA. In this case, I mis-typed the very last byte in the short program at address AA25. The data entered should have been 04 rather than 06. For those of you who did not catch it in the written instructions and entered the data as it appeared, found your WHITE SQUARE to be QUITE INACTIVE! The intent of the program was to display movement, with delay, from the top of the screen to the bottom.

MARCH-HELPFUL HINTS-WIRING FOR THE OLDER TYPE GEMINI 10 PRINTERS.

The lead designation should have been PIN 8 to PIN 8 rather than PIN 1 to 8.

CLARIFICATION

JULY ISSUE-PROGRAM "PAIRED ITEMS"

The article was weighted heavily with technical information about the operation of the program itself but should have included a brief description of what the program actually IS!

It could be considered a game, more specifically "A MEMORY GAME". The player enters a chosen number of PAIRS that are related to each other such as, STATES/CAPITOLS, POUNDS/KILOGRAMS, DOLLARS/ENGLISH POUNDS, etc. After all pairs are entered, the program will give you the first word of the pair and see if you can match it with its associated word that you entered initially. This can be recalled on a RANDOM or ORDERED basis. The computer will keep track of the % of correct entries. There are many variations that can be achieved and more than one player can be included.

CLARIFICATION

JUNE ISSUE-DATA HOP

As explained, the cable used with the VOLKSMODEM requires wiring changes. This is true only if an 'A' cable is ordered. We have just learned that if a 'B' cable is ordered with the modem, NO WIRING CHANGE IS NECESSARY. Be sure to specify the 'B' cable when ordering this modem.

ARE YOU MISSING AN ISSUE?

If so, please let us know which month and we'll send one out to you promptly!

MAY Q & A TOSS UP...NO RESPONSE TO DATE!!??
Please..take another look at the MAY issue Q & A TOSSUP section. If you could respond to ANY of the questions listed therein, we would ALL sincerely appreciate it!!

HELPFUL HINTS

This first HELPFUL HINT comes to us from FRED VANOEVEREN, Grand Rapids, MI. Fred writes: "I sent along a short program which can be used to convert a NUMBER to a STRING and back again. Since I keep all my bowling and golf league data as numbers, I have to use routines of this type to get strings for sorting. By concatenating the number strings with NAMES, DISK RECORD POINTERS, etc., I have been able to produce ORDERED LISTS for almost any type of data. Perhaps this routine would be of interest to others."

```
10  REM     ...CONVERT A THREE DIGIT DEC NUMBER TO A STRING
20  DIM A$(2): REM   ...DIMENSION A THREE CHARACTER STRING
30  INPUT "INPUT ANY NUMBER (0-999) ";N
40  N1=INT (N/100): REM    ...GET HUNDREDS
50  A$(0)= CHR$ (48+N1): REM   ...CONVERT TO STRING AND PLACE IN VARIABLE
60  N2= INT ((N-N1*100)/10): REM    ...GET TENS
70  A$(1)= CHR$ (48+N2): REM     ...TO STRING AND INTO VARIABLE
80  N3=N-N1*100-N2*10: REM     ...GET ONES UNIT
90  A$(2)= CHR$ (48+N3): REM    ...TO STRING AND INTO VARIABLE
95  PRINT "NUMBER ";N;"  IS NOW STRING ";A$
100 REM     ...THIS ROUTINE WILL CONVERT THE STRING BACK TO A NUMBER
110 N4=( ASC (A$(0))-48)*100
115 N5=( ASC (A$(1))-48)*10
120 N6=( ASC (A$(2))-48)
125 N7=N4+N5+N6
130 PRINT " ...AND AGAIN THE NUMBER ";N7
```

The next HELPFUL HINT was supplied by BILL BOWMAN, So. Yarmouth, MA. The program is an example of selecting items from a menu without using an INPUT prompt and RETURN KEY. Line 70 is the key to this function which is also explained in the APP BASIC LANGUAGE REFERENCE MANUAL. Thank you Bill for sending the program in.

```
5  GOSUB 4000
8  PRINT "PRESS KEY TO SETUP 'MENU SELECT'": PRINT "WITHOUT USING  INPUT & RETURN KEY"
10 PRINT : PRINT "LIST TO SEE HOW IT'S DONE  ": PRINT "RUN TO SEE EXAMPLE WORK"
9  PRINT : PRINT "EXAMPLE FOLLOWS:-"
10 DIM NUM$(8):A$(10)
20 POKE 16142=0 : POKE 0=16:10
40 PRINT "1-5 PRINT "2-" PRINT "3-"
50 PRINT "4-" PRINT "5-"
60 PRINT "SELECT (1-5)"
65 IF NUM=NULL THEN 60
70 POKE 9B?= ASC (NUM$):4+1 WAS/X "700#?300"
80 IF NUM="7" THEN 1000
82 IF NUM="6" THEN 2000
85 IF NUM="5" THEN 3000
90 IF NUM="9" THEN 3000
92 IF NUM="7" OR NUM>"2" IF NUM<"6" THEN 4000
95 IF NUM="9" OR NUM>"2" THEN 4000
99 IF NUM<>"?" THEN "WRONG KEY": GOTO 4000
1000 PRINT : PRINT "#1 EXECUTED"
1005 GOTO 4000
2000 PRINT : PRINT "#2 EXECUTED"
2005 GOTO 4000
3000 PRINT : PRINT "#3 EXECUTED"
3005 GOTO 4000
3050 PRINT : PRINT "USE...IT'S EASY"
3070 GOTO 4000
4000 FOR I=1 TO 300: NEXT
```
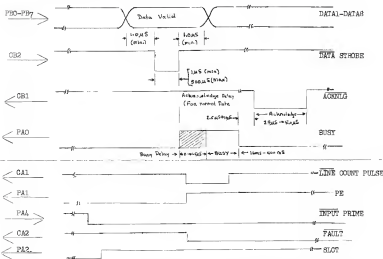
```
4005 GOTO 5
4010 CALL 17044: POKE 46>12: POKE 4096,1:0
4020 RETURN
```

The next two pages contain the remainder of the HARDWARE HINTS on CENTRONICS interfacing, supplied by DAVID PRESCOTT of A.I.T. Thank you David!

Schematic Diagrams:

Signals from IMPIA   Signals to Printer
Mass Term. Conn.   Centronics Conn.

The LED is used to indicate that the buffer circuit is getting power from the IM-1
through the IMPIA card. The two 74LS241 I.C.'s are used to buffer the higher current required
by the printer interface and thereby protect the parallel port inputs and outputs. Although
the tri-state of this chip is defeated by grounding the enables, it was chosen because of its
ease in straight through wiring and because it has eight drivers per chip saving space. The
0.1 uf caps provide glitch protection for the switching gates. Wire all the ground together
in daisy chain style so that there is a good signal ground between the IMPIA and the printer.
The +5 volt supply from the printer is not connected to anything. This is to prevent supply
voltage conflicts between the IM-1 and the printer. The I.C.'s supply pin 20 is connected to
the five volt supply brought out by the IMPIA edge connector.

Signal timing conditions for programming the parallel port.

PA0–PB7 — DATA1–DATA8
Data Valid

CB2 — DATA STROBE

CB1 — ACKNLG

PA0 — BUSY

CA1 — LINE COUNT PULSE
PA1 — PE
PA4 — INPUT PRIME
CA2 — FAULT
PA3 — SLOT

Set up the parallel port in the following manner. Data register B side all outputs. Control register B-side in data handshake strobe mode, CB2 strobe low, CB1 accept low pulse from ACKNLG. Because of the speed of the IM-1 clock the proper data set-up and strobe pulse durations should fall right within specifications. The acknowledge pulse from the printer will set the IRQBI bit in the B-side status register, so you know when to transmit the next character. Set up the A-side data direction register as PA0 - PA3 as inputs and PA4 as an output. PA4 will be used to output a low pulse INPUT Prime to clear and activate the printer. PA0-PA3 will accept the BUSY,PE,SLCT,and OGCLT, if you should desire to use them in software control. Set the A-side Control register with CA1 and CA2 as inputs with active low s s trigger on both. CA1 will then set IRQAI every time the printer does a line feed so that you can keep track of what line you are on, on your paper. CA2 will set The IRQA2 bit in the status reg. A should a fault occur on the printer.

Because the IM-1 does not have software to control a parallel printer interface it will be up to you to right the routine. It is most efficiently done in machine code, however it could be done in basic as well. Good luck and have fun.

All information in this article was addressed to the specifications supplied by Centronics Printer Manual Appendix B- Parallel Interface Specifications, to comply with the Centronics Interface characteristics, and matched to the INPIA parallel port. A.I.T. will not be held responsible for any errors which may occur herein nor any damage to either units caused by user wiring errors or abuse.

```
10 (DATACOMP COMPUTER SOFTWARE
20 (35272 STANTON POINT RD.
30 (INGLESIDE, IL 60041
40 POKE 36574,30
50 CALL 17044
60 POKE 49844,21: PRINT "        HAPPY DAZE "
70 SHAPE =4: FOR X=1 TO 255 COLOR =X
80 HLIN 0,21+X: VLIN 0+15+X: HLIN 0,31,15: VLIN X+15,31
90 FOR L=1 TO 504 NEXT : NEXT
100 MUSIC "1#4 4#0 1#0 +1#0 4 +0#4 4(#0#0"
150 MUSIC " +2 4#0 4#0 +1#0 4 +1#0 4#00#0"
160 MUSIC " 1(4 1#0 +1#0 +1#0 4 +4#0 4#0000#0"
170 MUSIC " +2 4#00 4#00#0 +1 4#00 4 +4#0000#0"
220 MUSIC " +1 #1 4#0# +1#0 4 +4#0000#0"
230 SHI C=1 TO 5
240 MUSIC " +4#0 4#0 +4#0 4#0 +4#00 4#0 4#0 4#00#0"
250 MUSIC " +4#0# 4#0 4#0 4 +4#00 +4#0000#0"
255 NEXT
260 MUSIC " 4 4#0 4 4#0# 1 +4#0 4"
270 MUSIC " +1 +1 +1 4#0 +1#0 4 4(#0#0#0"
280 MUSIC " 5#0 5 5#0 5#00# 7#0 7 +#00000#0"
300 MUSIC " +1#0 +1 4#0# +1#0 4 +4#0000#0"
310 MUSIC " +4#0# 4#0 +4#0 4#0 +4#00 4 4#0 4#00#0"
320 MUSIC " +4#0# 4#0# 4#0 +4#00#0"
```

```
10 FOR I=1 TO 2
12 MUSIC "2B0# #2B0500#500#0#80000     10000000#85000100000#0     10007000300500000000#0"
14 MUSIC " "
16 MUSIC "#80000000#0000000    2B00#CB0#CB00000#CB#00B0#0#2#40B00000"
18 MUSIC " "
20 MUSIC "#80000000#0000000    50B4000#B00500000000#0"
22 MUSIC " "
40 MUSIC "#80000000#B00#80000000    500#CB00#C0B0000#0"
42 MUSIC " "
44 MUSIC "B000#0B0000000B000B000    5000#780000#85000000000"
46 MUSIC " "
100 MUSIC "5B00#400B0#L0#500L00B00000"
110 MUSIC " "
120 MUSIC "5100#B000#L00B0000#00#    420B0#700B00 410B0#800L00000000"
130 MUSIC " "
140 MUSIC "#800#B0#20B000000000    50B4000#0 4200B000B000000000"
142 MUSIC " "
16 IF I=1 THEN NEXT
17 IF I=2 STOP
```

The upper program was supplied by RUSS WEXHAM
of DATACOMP SOFTWARE and is the theme from HAPPY
DAZE. The lower program is a sample of the many
tunes submitted by HAL DIXON. You'll recognize
the tune immediately!

SHORT PROGRAMS

At last, a base conversion program that allows you to add, subtract, multiply, and divide in decimal, hexadecimal, octal, or split octal and have the ability to retrieve the answer in ANY of these bases that you choose! OUR THANKS TO LOUIS DOOLEY FOR THIS PROGRAM.

LEGAL EXPRESSIONS:
"N" = any number between 0 and 65535
"*" = math operator: + - * /
"C" = number base code:
    T = DECIMAL (DEC) (default code) the program defaults to DECIMAL
    H = HEXADECIMAL (HEX)
    Q = OCTAL (OCT)
    S = SPLIT OCTAL (OCS)

FORMATS	EXAMPLE
N=N=	197+181=
N+=	+476=
N=C=	198I=H
CN=C=	H100=Q
Q	Q
N=N=C	4095+368O=H
CN+N=	Q377+617 377=N
CN=CN=	H74=Q
CN+N=N=	T1365+597=H
N=CN=	dD2+E512=
N=N=CN=	Fa+Q37-T
=N=	+59-8
=N=C=	+577 377=H
=CN=	+77N

(SPACE PERMITTED ONLY IN SPLIT-OCTAL NUMBER)

("OUT OF RANGE" print includes input or result greater than 65535 and improper characters) Note: Impractical to check for ALL possible errors in input; user must still be careful and precise.

SAMPLE TEST DATA:

DEC	= HEX	= OCT	= OCS
1234	04B2	2322	4323
4095	FFF	7777	17 377
7184	CF0	6160	14 160
3568	DF0	6760	15 360
3947	F6B	7553	17 153
65535	FFFF	177777	377 377

'BASIC' PROGRAM ROUTINES		
LINE	FUNCTION	EXPRESSION
1000	HEX TO DEC	(HS to T)
2000	DEC TO HEX	(T to HS)
3000	CALL 17846: POKE 40964,25 POKE 40964,0	
4000	DEC TO OCT	(T to OS)
5000	OCT TO DEC	(OS to T)
6000	STRING TO NUMBER	(S to N)
6000	NUMBER TO STRING	(T to NS)
300+	CALCULATION	(T and V)

```
10 DEL 17846: POKE 40964,CX: POKE 40964,0: POKE 24570,30
15 PRINT "NL BASE CONVERS, (L, DOOLEY)"
17 PRINT "      (FROM MEATH 85)": PRINT
20 PRINT "NE L(6)=T(DEC), H(HEX), Q(OCT), S(OCS)"
22 PRINT "NE L(6)=*(L*)+ (+)+ (-)+ (/)"
25 PRINT "EXPRESSION FORMATS:"
26 PRINT "NE L(6)=N, CN=H"
27 PRINT "NE L(6)=N (CN=,/CALC,)"
28 PRINT "NE L(6)=*N=N (CN=,/CALC,)": PRINT "NE L(6)=*C (CONV, CHAIN,)"
29 PRINT "NE L(6)=N (*=N , BETWEEN 0 & 65535)"
30 PRINT "NE L(6)=*C = *+ + *+ & /"
35 PRINT "NL MAY BE PRECEDED BY BASE CODE"
36 PRINT "NL MAY BE FOLLOWED BY BASE CODE"
37 PRINT "DEFAULT = T OR LAST-USED CODE"
38 FOR K=0 TO 1: PRINT
40 FOR I=1 TO SSMAX IF  KEY= '(8)='" THEN 55
51 NEXT
55 DIM IK(1),IK(3),IK(8):IK=41(41)
60 CALL 17846: POKE 40964,CX: POKE 40964,0
60 K=1:KK=952
61 K=K:C=1
62 KN=1:KK=0
63 INPUT "EXPRESSION ">GR
64 IK(8)=IK(8)
90 IF KK(8)=*" THEN  GK(1)=*DEC*: GOTO 110
100 IF KK(8)=*H* THEN  GK(1)=*HEX*: GOTO 110
110 IF KK(8)=*Q* THEN  GK(1)=*OCT*: GOTO 110
120 IF KK(8)=*S* THEN  GK(1)=*OCS*: GOTO 110
```

SHORT PROGRAMS

```
130 IF EN(A)="+" THEN M=1: GOTO 400
140 IF EN(A)="-" THEN M=2: GOTO 400
150 IF EN(A)="*" THEN M=3: GOTO 400
161 IF EN(A)="/" THEN M=4: GOTO 400
162 IF EN(A)="=" THEN 160
163 IF EN(A)="(" THEN PRINT "ERROR": END
164 IF EN(A)="P" THEN V=V: GOTO 400
165 IF EN(A)="Q" THEN 500
166 IF EN(A)="." OR EN(A)="0" THEN 500
540 C=C+1
170 N$(C)=EN(A)
185 IF M=0 THEN 191
190 ON M GOSUB 1000,2000,3000,3120,500
191 IF T>65535 THEN RETURN
200 A=A+1
201 GOTO C$(NE2)*2000,3000,4000,5000
301 IF E=1 THEN 500
302 PRINT TAB(23)"N$":TAB(4)B
310 T=T+N$(A): GOTO 500
310 T=T+V: RETURN
310 T=V+T: RETURN
310 T=V/T: RETURN
400 IF EN(A)="." OR B=CODE(500)*5000,6000,7000,800
410 M$(NE2)=C: GOTO 160
500 PRINT "OUT OF RANGE":"TEN=E=E": GOTO 5W
1000 T=E=E:E$=E$E:E$
1050 FOR X=1EN THEN 1
1060 ON B=CODE(N$(X)): IF X$P THEN V$=V-7
1610 IF X<P THEN V=V-1
1100 IF X<P THEN X$=1: RETURN
1050 IF T>65535 THEN E=1: RETURN
1050 B=B+1
1090 RETURN
1000 RETURN
2000 N$=N$+N$+N$+N$:V$=V-1
2005 IF T=E THEN E=1: RETURN
2010 IF T>65535 THEN E=1: RETURN
2020 V=V+1: RETURN
2030 H=H+N$(X-1)=V$:V=V+7
2040 H=H-V: IF X>1 THEN X$=H*7
2050 IF H=INT(H):T-1 THEN E=1: RETURN
3000 H=H+V
3030 RETURN

4000 N$=N$+E+E+N$=32768
4005 IF T>0 THEN E=1: RETURN
4010 IF T>65535 THEN E=16384
4020 H=INT (T/65535)
4010 H=INT (V$(X)+E-1)
4050 RETURN
4060 H=H+V: RETURN
4070 IF X<1 THEN 4080
4090 IF E(X1) THEN V$=V-7: RETURN
4090 ON T(X) THEN 4160
4100 IF E(T) THEN 4160
4110 H=H+V: RETURN
4200 FOR X=1 TO V STEP -1
4320 IF T(X)=0 THEN V$=V-7: RETURN
4010 IF INT(X)=INT(X)-T<1 THEN E=1: RETURN
4040 V$(X)=INT (T(X)-T(X))=V$
4050 IF X<P THEN RETURN
4060 V$=V/H
4070 V$(X)=INT T$(X)=T-V/H
4080 V$=V/H
4100 V$(X)=INT (T-V/H)
4100 V$=V+H-X-1: IF X=P THEN RETURN
4140 GOTO 4050
```

PRODUCT REVIEW

This fast action 8l ASS game has been around for quite awhile but there have been a few inquiries about it lately. The game begins by placing you in a maze with no exits. You maneuver a HI RES character around the playing field and try to destroy FOUR security power banks by running over them with your man.

Roving guards occupy the maze with you and will chase and fire bullets at you. You can fire back, and each guard that is destroyed is worth 10 points. If a guard runs over a spot where another guard was killed, a new guard is re-born and BOTH will chase you.

Guards in bunches are almost indestructable and your bullet will not usually penetrate them.

The walls that make up the maze are HIGHLY EXPLOSIVE. You may not fire more than one bullet at a time. Your bullet must hit something and explode before you are able to fire again.

This game has several other distinct features that make it very interesting to play. It's a game that requires quick responses along with a VERY WATCHFUL EYE. The game comes with an instruction sheet and will work on both color and black and white sets.

A few more features to mention are:

If a dead bullet is blocking your way, DON'T run into it or you may BLOW UP! First shoot the dead bullet then go on through it. After firing, if you run after your own bullet, it will have a better chance of penetrating a guards bullet and destroy him. Try to remember where you kill guards and try to keep the live ones away from that area. It's easy to trick the guards into running into your bullet.

When a guard moves next to you, you must run up or down to escape. If you move right or left, you will most likely get zapped by him!

Each security power bank that is destroyed is worth 100 points. Guards destroyed as mentioned are worth 10 points, and each NEW LEVEL that is achieved by eliminating guards in the proceeding level is worth an additional 1000 points. The game is written in MACHINE LANGUAGE for the play and BASIC for scoring. Some unique sounds are generated with the ML routines.

Note: The FRONT SCREEN (top left) indicates scoring provided in the game. Photo (bottom right) depicts the maze, power banks (objects that look like TV sets), the guards, YOUR MAN, and a few bullets (upper right).

NOT RECOMMENDED FOR THOSE WITH POOR PERIPHERAL VISION!!!!
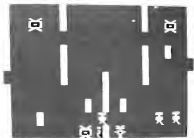
PRODUCT REVIEW

TAPE ASSEMBLER from HEXMART

If you have been following the ARCADE section over the past months and have been looking up each OP code to find its meaning, you have probably found the task quite burdensome and time consuming. Eric mentioned an ASSEMBLER in the beginning of his instruction but decided to introduce you to the INSTRUCTION SET and use IT for a better understanding of MACHINE LANGUAGE PROGRAMMING. This is a proven method of instruction.

The HEXMART TAPE ASSEMBLER allows a person who is somewhat familiar with machine language to write 6800 assembly language programs without having to spend HOURS looking through the instruction set tables for the right OP codes to use, or counting in HEX to figure out the branching instructions.

The TAPE ASSEMBLER is a LINE ASSEMBLER in that it assembles each line of 6800 assembly code as it is typed in. As an example, if you need to LOAD ACCUMULATOR A with a value, the HEXMART program will look up the OP code for you in its memory and enter this data at a particular address for you automatically. You would simply be required to type in LDAA with the value you want loaded, indicating the advanced mode desired. This makes it a very fast method of writing machine language programs because the end result is actually in MACHINE LANGUAGE.

You might consider this assembler to be INTERPRETIVE, that is, it takes an input such as LDAA IM and interprets the correct HEX value that corresponds to the instruction. This value is then placed in consecutive addresses almost immediately thus causing the entire process to move swiftly and accurately. An important point to remember is that success with this assembler, as with any other, is based primarily upon the USERS KNOWLEDGE of machine instruction and experience with it. The program is written in BASIC with a machine language routine that searches through the entire built in Instruction set table to find the correct OP code to use. The average time for the program to assemble a line of code is three to four seconds.

The TAPE ASSEMBLER is particularly useful for assembling relatively small programs but can also be used for larger ones since

it has capacity for 65 labels and 45 deferred operands in 8K.

The program comes with 27 pages of instruction that include an introduction to assembly language programming, how to use the assembler, work sheets, and information taken from the APP TECHNICAL REFERENCE MANUAL for further help in understanding assembly language programming. If you are serious about ML or assembly programming, the TAPE ASSEMBLER will take a lot of the chore out of the work. It is a common tool for virtually all programmers who work with machine language routines and programs.

Note! If you are just starting out in this area, you still need a good idea of how OP codes are used and what functions they perform. This is the intent of the ARCADE section. STAY WITH IT. In order to clearly understand Machine functions, the ARCADE should prove to be very helpful. After you have gotten a good grasp of programming directly, try the assembler and find out how much EASIER it makes the ENTIRE JOB!

WHEN ARCADE DATA WAS ARRIVED? The WICRON wheel is ready to go in order to get as much in this issue as possible. Eric begins this issue with BRANCHING techniques.

If the program appearing in last months issue was entered correctly you should see a square move smoothly across the screen from left to right and gradually from top to bottom. If not, check your inputs carefully and try again.

BRANCHING

The machine language branch statements are similar to BASIC IF THEN GO TO. You are checking for a certain condition (depending upon which branch statement is used) and then branching (or jumping) if that condition is met. If it isn't, the program continues on.

Branch statements can be used after any instruction but usually after a compare instruction or a subtract command. It can also be used after an INCREMENT or DECREMENT command as in our previous program example. Branching is best used when you need to go to a relatively close location. The limits one around 128 bytes in either direction from where the program currently is at the time. If your using a good ML ASSEMBLER your branches are automatically calculated. When using the APF monitor as we are in this instruction, you will have to count how far away you want to go by counting for yourself. If you branch wisht, for away from your current address, alot of your time will be spent counting bytes. I suggest using the following trick when you want to go to a far location on a certain condition. Let's assume that you want to branch if a certain number is greater then zero. It would look like this in machine code:

```
B1 COMPARE ACX.A IMMEDIATE
XX DATA TO COMPARE ACX.A TO
AB BRANCH IF LESS.. A=XX
03 WHERE TO BRANCH TO
XX PROGRAM CONTINUES
XX
```

In the example you will be branching 03 bytes FORWARD in the program from where you are. You can change the branch if greater then ( ) 0 to branch if less then ( ) 0. In this way if it is greater then 0 you SKIP and go to a JUMP instead. Look at the following example?

```
A1 COMPARE ACX.A IMMEDIATE
XX
XX BRANCH IF LESS THAN 0
03 WHERE TO BRANCH TO
XX JUMP
XX TO THE LOCATION
XX IN BYTES AWAY
```

This second example will save you time counting all those bytes!

We can branch forward and backward. Branch instructions are TWO bytes long, the second byte is called the OFFSET byte. This byte determines how far away we will branch, forward or backward. To branch forward just count the number of bytes to where you want to go. You DO NOT count the offset byte and the byte after it in number 00. Then comes 01, 02..... etc. You can see how we did this in our example with the moving square. At A410 we wanted to branch over the return code or 39, so we used in offset value of 01. We can only branch forward to around 3F because to branch backwards we start at the offset of FF and count backwards FF, FE, FD, FC... We did this in our example at step A410. We wanted to branch back to A410 or back 7 bytes, so we used an offset of FB. For convenience, the BRANCH CHART included in this issue should be helpful to you to figure your branching values.

Try experimenting with the moving square example. By changing the delay variable, you can get a real good feel of how fast ML is. There are two variables in accumulator A that are most effective. Start by changing the value at A40F from FF to 7F. This is about HALF THE DELAY from the original version. Do this by using the ML monitor command MW A40F 7F 7F. Now, run the program. Notice the speed change from the original. Try changing A40F to 3F. Then try 01. You will get to a point where it goes so fast across the screen that it no longer looks like the square is moving. It will appear to jump from one place to another.

JOYSTICKS

If you have ever written any games in BASIC, you know one of the most time consuming things to do is to READ the joysticks during the program operation. This problem is easily overcome using machine language. To make things easier, I've included a routine in the ROM that reads the joysticks and places the results at an address. To use these routines we simply JUMP to them using a JUMP TO SUB-ROUTINE command. When the subroutine is done, it will automatically return to wherever it left off in the program.

To read the LEFT joystick, you would jump to the subroutine at 6F0E. Your computer now looks at the joystick. If anything is pressed, a BIT in the CONDITION CODE REGISTER called the CARRY BIT will be SET. This will be explained later. It also READS what was pressed on the joystick and stores it at AA11 CODE for this value at address 01F2. Our program would then look at 01F2 to see what was pressed.

THE ARCADE

This is how we would use this subroutine:

First, clear the carry bit using the OP code 0C which is CLEAR CARRY. This is done to reset the joystick in case it was already moved once. We then jump to the joystick ROM routine we mentioned. When we come back from the ROM routine we use the OP code to test if the carry bit is set, or (25 in HEX). If we used the joystick, we would branch to another routine that reads location 01F2 to see which way the joystick was moved. If we wanted to read the RIGHT joystick, we would do the same thing, only JUMP to a different ROM routine which is located at 4109. The result would still be found at location 01F2. The result at location 01F2 is the ASCII equivalent. The values that appear at location 01F2 will be as follows:

```
NORTH = 4E
SOUTH = 53
EAST  = 45
WEST  = 57
FIRE  = 21
```

We can also use a ROM routine to get a key input from the MAIN KEYBOARD. It will not return a shifted keyword such as the control key and top 2 rows. To use it simply jump to the subroutine at 80CF. When the program returns from the subroutine, the ASCII code for the key pressed will be in ACCUMULATOR A. If ACCUMULATOR A equals 0, then NO KEYS WERE PRESSED.

Next on Eric's agenda is a 150 byte machine language program that moves a white square using the joystick with the ROM routines. There is not enough room in this issue to accomodate the data, so, a small portion will appear herein and the remainder in next months issue.

For this month, enter only the BASIC program that appears below and CSAVE this information so tape so that it can be retrieved easily for next month's issue.

The first 3 BASIC lines contain 'A' characters after the REM statements. Enter the amount of A's shown after the REM.

```
10REM (total of 51 letter A's)
15REM (total of 51 letter A's)
20REM (total of 50 letter A's)
25POKE 24578,38
30CALL17045
40CALL42012
45CALL42012
50GOTO45
```

BRANCH SHORT

USE TOP AND LEFT FOR BACKWARD BRANCH
USE BOTTOM AND RIGHT FOR FORWARD BRANCH
COUNT THE NUMBER OF BYTES TO THE
TARGET INSTRUCTION...THEN
GO TO THE CHART

EXAMPLE OF BACKWARD BRANCH:

```
 9  8  7  6  5  4  3  2  1
86 23 A7 00 BE 00 C6 00 27 06
# of BYTES = 9   LEFT = F7
```

EXAMPLE OF FORWARD BRANCH:
(Note where count begins)

```
86 33 A7 00
  | 1  2  3  4  5  6  7 |
86 23 A7 00 C6 07 00 00 27 06
# of BYTES = 6   BOTTOM & RIGHT = 06
EXAMPLES: FORWARD 122 = 7A
          BACKWARD 111 = 91
          FORWARD  33 = 21
```

DATA SHOP

Briefly, let's examine the last form of IM-1 to IM-1 communication...NEITHER IM-1 RUNNING A TERMINAL EMULATOR PROGRAM.

After the two computers are connected via modems, the transmitting and configures his IM-1 to PRINT TO THE LINE by typing in the statement 'PRINT-1. The receiving and relinquishes control of his IM-1 by positioning it in the PRINT-2 mode. At this point, the transmitting end has full control over the receiving end IM-1.

Basing our experimentation with this form of communication we found that normal PRINT and CONTROL statement can be sent from the distant end to the IMMEDIATE MODE ONLY, that is, the statements must be typed by the transmitting and without any program operation or interface. For example, to print the word HELLO, the transmitting end (in the PRINT-1 condition as mentioned previously) would merely type PRINT"HELLO" with a carriage return. The distant end would receive the word HELLO on his screen, but, because the word in SPOILED by itself, his IM-1 will respond with the word WHAT. This is of no consequence when single sentence conversation is desired in this mode. Transmitting VALID statements as PROGRAM DATA requires a little but more work and will be explained in detail.

Virtually ALL BASIC commands can be sent to the receiving end as long as they are included with a PRINT STATEMENT. Commands such as (SAVE, CLOAD, POKE, RUN, VLIN, can also be transmitted to move information stored in the receiving ends memory, or to produce graphics on his screen. For example, to draw a horizontal line across the receiving end IM-1, the transmitting end would type PRINT"HLIN1,30,1". Only the command HLIN1,30,1 would be sent out to the receiving end. His IM-1 would interpret this command as a VALID statement and would produce a line across his screen providing the SHAPE and COLOR was established previously. To the receiving IM-1, this command would seem to have been from the keyboard directly, however, it produces the same results when coming in from the line. It seems a bit EERIE to have someone else CONTROLLING YOUR computer!

If a musical score is to be sent for IMMEDIATE LISTENING, the transmitting IM-1 would include the music within a special print statement such as PRINT"MUSIC"123"". Notice the use of an apostrophe at the beginning and end of the above statement. This allows the needed QUOTATION MARKS to accompany the MUSIC statement so that the receiving IM-1 accepts the command MUSIC"123". Any apostrophe will be eliminated from the transmitted data. This arrangement should also be used when sending line by line program data whenever QUOTES are needed. See example below.

As you can plainly see, there are many things that CAN be accomplished in this mode, but, in an extremely SLOW fashion. For transmitting program data (directly typed), it is much easier to use a terminal emulator program on the transmitting end.

NEXT MONTH! The conclusion of this article and a look at a few of the popular BULLETIN BOARD SYSTEMS throughout the country using ITEM-1 with a printer bridged on the line at the same time. See you then!

SAMPLE PROGRAM THAT CAN BE TRANSMITTED TO THE DISTANT END, LINE BY LINE, FROM AN ON TWO DISTANT END COMPUTER.

```
PRINT"10PRINT"ITS TIME TO WAKE UP THERE""
PRINT"20MUSIC"555566655444555""
PRINT"30POKE-173050:MEXT"
PRINT"40CALL17046"
PRINT"50GOTO10"
PRINT"RUN"
```

Note! As you type, the program will be entered into the distant IM-1's memory. The command RUN will cause the program to begin running so the distant end? To give control back to the receiving end, type the command PRINT"PRINT-0". The distant end will then be able to CSAVE the program and have full control once again.

MICROPROCESSOR GLOSSARY

ACCUMULATOR The register where arithmetic or logic results are held. Most MPU instructions manipulate or test the accumulator contents.

ACCESS TIME Time take for specific byte of storage to be delivered to processor.

ACIA Asynchronous Communications Interface Adaptor — interface between asynchronous peripheral and MPU.

A/D Analogue to Digital. The part of the MPU where arithmetic and logic functions are performed.

ADC Analogue to Digital Converter.

ADDRESS Code for specific memory location. Every code location, word or byte, must have a unique address in computer's memory.

ADDRESS/REGISTER Register containing address of memory word being referenced.

ADDRESS BUS The bus carrying address of memory location being referenced.

ALGORITHM A set procedure for the solution of a problem.

ALPHANUMERIC Data that is a combination of letters (alpha) and numbers (numeric).

ALU Arithmetic and Logic Unit.

ASCII American Standard Code for Information Interchange.

ASSEMBLER A program to convert symbolic language instructions into machine language.

ASSEMBLY LANGUAGE Mnemonic programming language.

ASYNCHRONOUS Operations that proceed in uncontrolled time relationship.

BASE Register in MPU holding address of memory location.

BAUD Measure of speed of transmission (one Baud equals one bit per second).

BCD Binary Coded Decimal. Mode of operation representing decimal numbers where each 4-bit group represents one decimal digit.

BENCHMARK A common test for the comparison of which programs or hardware are judged.

BINARY Number system with a base of 2.

BIT Binary digit, 0 or 1. The smallest unit of information in a binary system.

BRANCH Program address of an instruction will be halted in time relating to next instruction.

BUFFER Circuit to provide isolation between different sections of a system and store data so it can be passed at a different rate.

BUS A program that moves the program to a location in the computer store.

BYTE A group of bits, usually 8, forming a unit of information.

CLOCK The basic timing in a MPU chip.

CONTROL The section which controls logic and timing signals needed for the MPU.

CPU Central Processing Unit, the MPU which is heart of the system.

DATA Information in digital form.

DATA BUS Bus carrying data between memory, CPU and peripherals.

DECIMAL Number system with a base of 10.

D/A Digital to Analogue.

DIGITAL Representation in discrete form.

DUPLEX Transmission of data in both directions.

EDIT Program to prepare a source program for the compiler.

EPROM Erasable Programmable Read Only Memory.

EXECUTION TIME The time taken to perform an instruction.

FIRMWARE Instructions or data permanently stored in ROM.

FLAG A bit (or flip-flop) that may be set or reset under software control.

FLIP FLOP Basic storage element.

FLOPPY DISK Mass storage of which makes use of a flexible or floppy magnetic disc.

FLOW CHART A diagram representing the logic of a program.

GATE Basic logic element.

HALF DUPLEX Data transfer in one direction at a time.

HAND SHAKE Sequence of data transfer between CPU and peripheral whereby CPU asks peripheral if it is ready to transfer data.

HARD COPY Output in a permanent, visible form determining the overall logic operation.

HARDWARE The physical components of a system.

HEX Hexadecimal. Number system with a base of 16.

INDEX REGISTER Register whose contents are added to the address part of certain instructions.

INPUT/OUTPUT The process that deals with transfer of data to and from the CPU and peripherals.

INTERFACE Device matching characteristics to allow transfer of data between systems.

INTERRUPT An external signal that causes the MPU to stop present program and go to subroutine.

I/O Input/Output.

LABEL A symbolic name used to identify an instruction or location in a program.

LATCH Element to hold a value.

LOOP A sequence of instructions executed repeatedly until a certain condition is met.

MACHINE CODE The binary code used by the processor.

MACRO A group of instructions represented by a single name.

MEMORY A device used to store digital information.

MICROCOMPUTER A computer built around a microprocessor.

MICROPROCESSOR A CPU implemented by one or more large scale integrated circuits.

MNEMONIC A word or group of letters used to represent an instruction.

MODEM Modulator/demodulator used to send and receive data over phone lines.

MONITOR A control program.

MPU Microprocessing Unit.

NON-VOLATILE Memory which will retain data when the power supply is turned off.

OBJECT CODE The output of an assembler or compiler, in machine code.

OCTAL Base 8 number system.

OP CODE Operation code, part of instruction specifying operation to be performed.

OPERAND The data operated on by an instruction.

PARITY A bit added to data to make total of 1 bits always odd or even, used to detect errors.

PERIPHERAL Equipment for input or output of data from a system.

PROGRAM Set of instructions to solve a problem.

PROM Programmable Read Only Memory.

RAM Random Access Memory.

REGISTER A device for temporary storage.

ROM Read Only Memory.

SOFTWARE The programs used in a system.

SUBROUTINE A program that may be used repeatedly.

VOLATILE Memory that loses its contents when power is removed.

WORD A group of bits treated as a unit.

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

                    HARDWARE/SOFTWARE UNBELIEVABLE PRICES!!!!

OFFER #

   1    PAK-MAN, SUPERFROG, ESCAPE!! and 2 FREE APP PROGRAMS   ALL 5 ONLY  $8.50 shipped free

   2    MS. PAK-MAN, FOOSEBALL!! and 2 FREE APP PROGRAMS       ALL 4 ONLY  $8.50 shipped free

   3    ANY 2 OF MINE and 3 APP PROGRAMS ONLY $7.50 shipped free

   4    2 BASIC CARTRIDGES $23.00 EACH

   5    8K EXPANSION CARTRIDGES $20.00 shipped free...ONLY 2 LEFT IN STOCK

   6    AS IS IN-1's INCLUDES TOP, BOTTOM, 2 CONNECTOR $35.00 shipped free (CANADIANS ADD $5 s/h)

   7    APP DIAGNOSTIC and FREE APP PROGRAM $5.00

   8    ANY 3 OF MY GAMES: PAK-MAN, SUPERFROG, ESCAPE, MS. PAK, FOOSEBALL ONLY $5.00 INCLUDES FREE APP

        I JUST RECEIVED A NEW SHIPMENT OF SOFTWARE, SO IF I WAS PREVIOUSLY OUT OF A GAME YOU
        ORDERED, IT'S IN NOW. PLEASE LIST ALTERNATES, THESE GAMES ARE  G O I N G  F A S T!!!!!

   9    7 APP PROGRAMS ONLY $5.50 shipped free

  10    12 APP PROGRAMS ONLY $10.50 shipped free


   ORDER FORM          MAKE CHECK PAYABLE TO: ERIC BECKETT
                                              8836 W. Waterford Sq. S.
   OFFER #   PRICE                            Greenfield, WI  53228

        1-------$8.50                         APP PROGRAMS AVAILABLE
        2-------$8.50
        3-------$7.50             ELECTRONIC FILES         MUSIC COMPOSER
        4-------$25.00ea.         TYPING TUTOR             SPACE DESTROYERS
        5-------$20.00ea.         BUDGET MANAGER           BARSMAN
        6-------$35.00            PERSONAL BUS. MACH.       SHOOTING GALLERY
        7-------$5.00             BILLBOARD                CASINO
        8-------$5.00             SPACE, SIZE, SURFACE     BASEBALL
        9-------$5.50             MATH TUTOR               BOXING
       10-------$10.30            THE WORD FACTORY         BACKGAMMON
                                  SPELLING DUEL            CANTEENA
   TOTAL  $                                                JUMBLED UP THINGS

   CANADIAN ORDERS PLEASE
   INCLUDE $2.00 TO HELP WITH       Please list 'A' for ALL ALTERNATES
   SHIPPING.                        ___ PAK MAN    ___ SUPERFROG    ___ ESCAPE
                                    ___ MS. PAK    ___ FOOSEBALL

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```